

## ソフトウェア演習 II 問題 8

14S1020U 小林佐保

2016/11/16

## 1 解いた問題

1. 12 章の練習問題を、1～17 から 2 つ以上（似てないものを）選んで解け。

1. 基本手続き\*の追加
2. 基本手続き/の追加
3. 基本手続き-の改良
4. 基本手続き-の改良
8. 基本手続き list?の追加

2.18. 12 章の練習問題を、18～26、31、32 から 1 つ以上選んで解け。（文字列型の追加）

3.27. 12 章の練習問題の 27 を解け。（define の改良。）

5.29. 12 章の練習問題の 29 を解け。（let の実現。）

## 2 プログラム解説書

### 2.1 1.1. 基本手続き\*の追加

#### 2.1.1 ts:intern-primitive-procedures に\*を追加

これによって、(ts) 実行時に\*と関数 tsp:\*、引数の数 2 の情報が基本手続き\*として登録される。

#### 2.1.2 tsp:\*の定義

TS の integer として与えられた引数から scheme の値を取り出し、それらの引数を scheme の\*に与えて評価した値を TS の integer に直す。これをこの関数を評価した値とする。

### 2.2 1.2. 基本手続き/の追加

#### 2.2.1 ts:intern-primitive-procedures に/を追加

これによって、(ts) 実行時に/と関数 tsp:/、引数の数 2 の情報が基本手続き/として登録される。

### 2.2.2 tsp:\*の定義

TS の integer として与えられた引数から scheme の値を取り出し、それらの引数を scheme の quotient に与えて評価した値を TS の integer に直す。これをこの関数を評価した値とする。scheme の関数 quotient を評価した値は整数割り算の商であり、これは実数の範囲で割り算をし、小数点を切り捨てた値と同じである。

## 2.3 1.3. 基本手続き-の改良

### 2.3.1 ts:intern-primitive-procedures の-を変更

これによって、(ts) 実行時に登録される基本手続き-の引数の数を 2 から 'any に変更できる。

### 2.3.2 tsp:-の定義

再帰呼び出しによって和を求める関数 loop-sum を用意し、それを利用して、第一引数から第二引数以降を loop-sum に渡して得た和を引いた値を tsp:- を評価した値とする。

## 2.4 1.4. 基本手続き-の改良

### 2.4.1 tsp:-の定義

引数がひとつかどうかを判定し、二つ以上であれば 1.3. のプログラムを実行し、ひとつであれば、その引数から得た scheme の値を scheme の-関数に渡した値を TS の integer に直し、この関数を評価した値とする。

## 2.5 1.8. 基本手続き list?の追加

### 2.5.1 ts:intern-primitive-procedures に list?を追加

これによって、(ts) 実行時に 'LIST? と関数 tsp:list?、引数の数 1 の情報が基本手続き list? として登録される。

### 2.5.2 tsp:list?の定義

再帰呼び出しによって cdr を辿り、空リスト '() に行き着けば list である、そうでなければ list でないとする。

## 2.6 2.18. 文字列型の追加

### 2.6.1 ts:make-string 関数

scheme の値と STRING タグとのペアを作る。

### 2.6.2 ts:string?関数

STRING タグがついていれば string であると判定する。

### 2.6.3 ts:scheme-obj->ts-obj に string を追加

読み込み等するときに scheme の string 型であれば TS の string 型に変換するように追加。

### 2.6.4 ts:print-value に string を追加

TS の string 型は値をそのまま表示するように if の条件に追加。

### 2.6.5 ts:eval に string を追加

string 型を評価した場合それ自身を返すように追加。

### 2.6.6 ts:intern-primitive-procedures に代表的な文字列を扱う基本手続きを追加

今回は string-length、string=?、string-append を追加。

### 2.6.7 tsp:string-length の定義

scheme の string-length を利用して実装。

### 2.6.8 tsp:string=?の定義

scheme の string=?を利用して実装。

### 2.6.9 tsp:string-append の定義

scheme の string-append を利用して実装。

## 2.7 3.27. define の改良

### 2.7.1 ts:do-special-form 関数

(DEFINE) の case を編集。ts:do-special-form に渡された引数のリストの 2 つめの要素 (以下 define の第一引数と呼ぶ) がペア (要素 1 以上のリストもしくはペア) であれば、簡略記法であると判断する。その場合は、適切に引数の順序を入れ替え、等価な通常の記法の形に直して define-var に渡す。TS の lambda はそれ自身独立した関数として定義されていなかったため、ts:make-compound-procedure を利用した。簡略記法でない場合は以前の通りに評価する。

## 2.8 5.29. let の追加

### 2.8.1 ts:map 関数

let を実装するために、関数とリストを引数にとり、リストの各項目に関数を適応して得られた値をリストにして返す map 関数を実装。

### 2.8.2 ts:special-form?関数に追加

let を特殊形式として登録。

### 2.8.3 ts:do-special-form 関数

let に与えられた引数を map 関数を利用して適切に並べ替え、3.27. の define の改良同様、lambda の代わりに ts:make-compound-procedure に渡した。